Construction automatique d'adaptateurs guidée par une ontologie pour l'intégration de sources et de données XML

Chantal Reynaud, Brigitte Safar

Construction automatique d'adaptateurs guidée par une ontologie pour l'intégration de sources et de données XML

Chantal Reynaud[*,**] **– Brigitte Safar**[*,**]

[*] *LRI ; Univ. Paris-Sud, CNRS ;*
Bât. 490, 91405 Orsay, France
[**] *INRIA Saclay – Île-de-France ; Projet Gemo*
Parc Orsay Université
4 rue Jacques Monod; Bât. G
91893 Orsay, France
{chantal.reynaud,safar}@lri.fr

RÉSUMÉ. *Le travail décrit dans ce papier porte sur l'intégration de sources d'informations hétérogènes XML au sein d'un serveur d'information selon une approche mixte combinant médiation et entrepôt de données. Ce serveur dispose d'un schéma, ou ontologie, utilisé pour l'accès tant aux sources externes qu'aux données locales. La méthode que nous proposons est une méthode unifiée qui s'appuie sur l'ontologie et permet de réaliser à la fois l'intégration de sources et de données. Notre contribution est double. Elle porte d'une part sur la génération automatisée de mises en correspondance, ou mappings, entre l'ontologie et une nouvelle source à intégrer, d'autre part sur la construction automatique d'adaptateurs (wrappers en anglais) allant de la description du contenu abstrait de cette nouvelle source jusqu'à l'extraction des données. Des expérimentations ont été réalisées sur des données réelles dans le domaine du tourisme.*

ABSTRACT. *This paper deals with integration of XML heterogeneous information sources into an information server according to an approach combining mediation and data warehousing. A schema, or ontology, is used to access to the external sources and also to the local data. We propose a unified method based on such an ontology able to achieve the two kinds of integration. Our contribution is twofold. First, we propose techniques to automate the generation of mappings between the ontology and a new source. Second, we present an approach to automate the construction of wrappers starting from the description of the abstract content of a source and ending by data extraction. Experiments on real data in the tourism domain have been achieved. Analysis and comments of the results are given.*

MOTS-CLÉS : *intégration de sources hétérogènes, médiation, entrepôt, ontologie, mappings, adaptateurs, documents XML.*

KEYWORDS: *integration of heterogeneous sources, mediation, data warehouse, ontology, mappings, wrappers, XML documents.*

1. Introduction

L'intégration d'information est une problématique importante du fait du nombre croissant de sources d'information disponibles via le Web. Appliquée à un serveur d'information, elle donne l'impression à un utilisateur qu'il interroge un système unique et centralisé grâce à une interface d'interrogation uniforme basée sur un modèle du domaine d'application, l'ontologie. Il existe deux approches d'intégration : l'approche entrepôt de données et l'approche médiateur. L'approche entrepôt de données consiste à voir l'intégration comme la construction de bases de données sur lesquelles l'utilisateur posera directement ses requêtes. Dans l'approche médiateur, les données restent au contraire stockées dans leur source d'origine. Le système dispose en revanche de descriptions abstraites du contenu des sources et détermine, à partir de celles-ci, les sources pertinentes pour répondre à une requête.

Ce travail porte sur la construction d'un serveur d'information intégrant des sources externes tout en regroupant des données provenant de sources à l'origine externes, dans un entrepôt. L'objectif d'un tel système est d'être capable de fournir des réponses à des requêtes utilisateur exprimées dans un certain langage et à l'aide d'un certain vocabulaire, en combinant des éléments de réponse puisés dans des sources d'information externes, basées sur d'autres langages ou vocabulaires, ou dans l'entrepôt de données disponible en interne. La construction d'un tel serveur pose à la fois des problèmes d'intégration de sources, analogues aux problèmes rencontrés dans les systèmes médiateurs, et des problèmes d'intégration de données, similaires à ceux rencontrés lors de la construction d'entrepôts de données. Un schéma, ou ontologie, fournit le vocabulaire pour interroger le système, il établit également une connexion entre les sources externes intégrées et facilite l'accès aux données de l'entrepôt.

Etant donné ce contexte, nous proposons une méthode unifiée permettant à la fois l'intégration d'une source externe via une approche de médiation et l'intégration de ses données dans un entrepôt local. Les sources à intégrer que nous considérons sont des sources XML structurées, dans la mesure du possible, à l'aide du vocabulaire de l'ontologie du serveur d'information. Néanmoins, chaque source a une structure qui lui est propre et qui ne correspond pas forcément à la structure des autres sources déjà intégrées. L'intégration d'une nouvelle source n'est donc pas immédiate. La méthode que nous proposons se déroule en deux étapes. La première, semi-automatique, est une phase de découverte de mises en correspondance entre la source considérée (source contenant les données à extraire ou source externe à intégrer sans rapatriement des données) et l'ontologie du domaine du serveur. La seconde, totalement automatique, consiste (1) à construire une description abstraite du contenu de la source puis (2) à en extraire les données directement dans le format de l'entrepôt, l'extraction ayant uniquement lieu lorsque l'objectif est d'intégrer les données dans l'entrepôt du serveur. Ainsi, notre contribution est double. Elle porte d'une part sur la génération automatisée de mises en correspondance, ou mappings, entre l'ontologie et une nouvelle source à intégrer, d'autre part sur la construction

automatique d'adaptateurs (ou wrappers) permettant d'extraire les données d'une source pour les représenter dans l'entrepôt local. Le processus mis en œuvre permet également, s'il n'est exécuté que partiellement, d'intégrer une source externe selon une approche de médiation. Un prototype, *DétectHybrideMap*, a été développé et utilisé pour réaliser des expérimentations.

Ce papier est organisé de la façon suivante. Dans une première section, nous décrivons l'approche générale. La section 3 présente l'environnement de travail en définissant l'ontologie, l'entrepôt local, les sources et les mappings. La section 4 est consacrée à la découverte automatisée des mappings. La section 5 présente les deux phases de l'extraction des données, l'identification du contenu des sources sous la forme de vues puis la construction des wrappers proprement dits à l'aide de requêtes XQuery. La section 6 décrit les expérimentations concernant la découverte de mappings et réalisées avec le système *DétectHybrideMap*, puis la section 7, quelques travaux proches. Enfin, nous concluons et présentons quelques perspectives.

2. Approche générale

Ce travail se situe dans le cadre du projet PICSEL3[1]. Ce projet a pour objectif de poursuivre l'extension de la boîte à outils PICSEL (Rousset *et al.*, 2002) de façon à faciliter le déploiement de la plate-forme de médiation. Il combine une approche médiateur avec une approche entrepôts de données. Une telle architecture flexible permet de prendre en compte des besoins applicatifs fréquemment rencontrés quand certains fournisseurs de contenus délèguent le traitement de leurs données.

L'intégration de sources ou de données, quelle que soit la méthode, repose nécessairement sur l'exploitation de mises en correspondance entre les schémas des sources intégrées et l'ontologie du serveur. Nous souhaitons une approche de découverte de ces mises en correspondance qui soit générique, c'est-à-dire utilisable quel que soit le domaine d'application. Le processus que nous proposons se veut, par ailleurs, être le plus automatisé possible. Le concepteur ne doit avoir à intervenir qu'en fin de processus pour valider les résultats trouvés. L'approche de découverte de mappings que nous proposons satisfait ces contraintes. Elle est guidée par l'ontologie du domaine et s'appuie sur les mappings déjà établis entre l'ontologie et les sources déjà intégrées au sein du serveur.

Le problème de la construction d'un adaptateur, ou wrapper, dans le cadre du projet PICSEL3 peut être décrit de la façon suivante. Etant donnée *S* une source XML contenant un ensemble de données, déterminer le processus logiciel permettant de peupler *E*, l'entrepôt RDF local de PICSEL3 avec les données de *S*. Les contraintes à respecter sont les suivantes. Le module logiciel proposé doit être totalement intégré

[1] Projet ayant fait l'objet d'une convention de recherche (CRE) avec France-Telecom R&D (2005-2008). http://www.lri.fr/~sais/picsel3/

à l'approche PICSEL. Il doit tenir compte du fait que les sources sont évolutives. Régulièrement de nouvelles sources doivent pouvoir être intégrées au système. Par ailleurs, les données elles-mêmes évoluent. Il peut alors être nécessaire de ré-extraire régulièrement les données de sources dont le contenu a déjà été extrait. Un objectif important est donc l'automatisation afin de rendre la construction d'adaptateurs très simple et rapide. Enfin, l'approche doit, là aussi, être générique, applicable à toute source XML quelle que soit sa DTD. L'objectif est d'être capable d'extraire les données de n'importe quelle source dont le domaine d'application correspond à celui du système d'intégration.

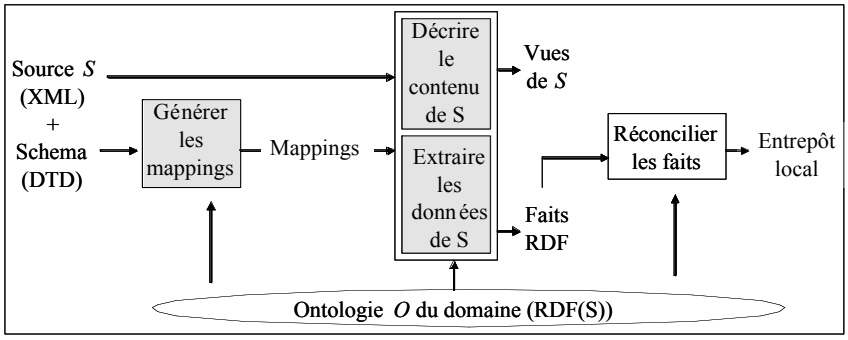


Figure 1. *Composants pour l'intégration d'une source et de données XML*

Le schéma de la figure 1 présente les différents modules conçus dans le cadre du projet pour intégrer le plus automatiquement possible des sources ou des données. Cet article se focalise sur les modules de génération de mappings, de description du contenu d'une source et de l'extraction des données. Ces traitements sont complétés au sein du projet par un processus de réconciliation de références permettant l'intégration de données externes dans l'entrepôt local en éliminant les redondances (Saïs *et al*., 2007).

3. Environnement de travail

Dans cette partie, nous décrivons l'ontologie, l'entrepôt local des données, les sources XML et les mappings.

3.1. *L'ontologie*

L'ontologie est composée d'un ensemble de classes ayant des propriétés et d'un ensemble de relations de subsumption ou du domaine entre classes. Une ontologie O est définie par un tuple (C, R) où $C = \{c_1, c_2, ..., c_n\}$ est l'ensemble des classes et où $R = \{r_1, r_2, ..., r_m\}$ regroupe l'ensemble des relations entre classes (R_1) et les propriétés des classes (R_2). Une relation r_1 de R_1 met en relation deux noms de

classes, ($\forall r_1$ (t_1,t_2), $r_1 \in R_1 \Rightarrow t_1 \in C$, $t_2 \in C$), et une propriété r_2 de R_2 relie un nom de classe à un littéral, ($\forall r_2$ (t_1,t_2), $r_2 \in R_2 \Rightarrow t_1 \in C$, où t_2 est un littéral). Dans le cadre du projet PICSEL3, le formalisme de représentation des connaissances adopté pour représenter l'ontologie est RDF-S (http://www.w3.org/TR/rd-schema/) dont les primitives permettent la représentation de classes, de propriétés, de hiérarchies de classes et de hiérarchies de propriétés. Il s'écrit à l'aide de triplets RDF (http://www.w3.org/RDF) de la forme <sujet, prédicat, objet>. Ainsi les relations r (t_1,t_2) sont traduites en RDF-S sous la forme de triplet (t_1,r ,t_2).

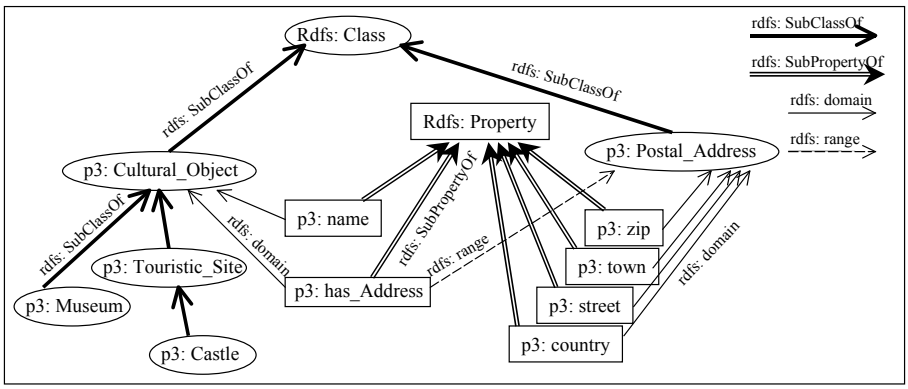


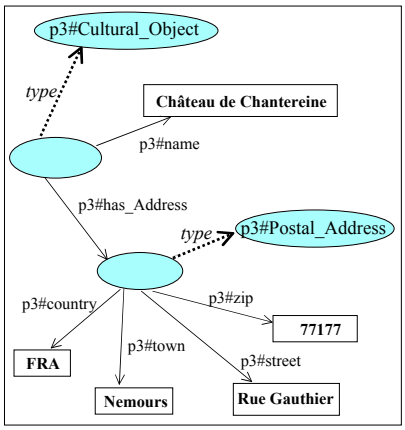
Figure 2. *Extrait de l'ontologie du domaine*

La figure 2 présente un extrait de l'ontologie du domaine considérée dans cet article, représentée dans le formalisme RDF-S. L'ontologie complète représentée sous forme d'un diagramme de classes UML figure en annexe. Dans ce graphe, les sommets ovales représentent les concepts, et les rectangles, les propriétés. Les étiquettes des arcs décrivent les relations RDF-S entre les sommets. Ainsi parmi les propriétés ayant pour « domain » la classe *Cultural_Object*, la propriété *has-Address* a pour « range » la classe *Postal-Address* et représente une relation dans notre ontologie, alors que *name* est une simple propriété. Les arcs RDF-S de type « range » issus de propriétés et pointant sur des littéraux ne sont pas représentés. Des informations sur les cardinalités des relations sont par ailleurs stockées car elles sont utiles pour la construction des adaptateurs (cf. section 5). Non représentables en RDF-S, elles ne figurent pas sur l'extrait de l'ontologie présentée figure 2.

3.2. *L'entrepôt local*

L'entrepôt local de PICSEL3 se compose d'un ensemble de faits RDF que l'on peut représenter sous forme de graphes. La figure 3a présente un exemple de graphe RDF décrivant une instance de *Cultural_Object* nommée *Château de Chantereine* et son adresse associée. Dans ce graphe, les sommets ovales représentent des concepts, et les rectangles, des littéraux. Les étiquettes des arcs sont de deux types : ceux précédés du radical *p3#* représentent des noms de propriétés ou de relations issus de

l'ontologie (*p3* pour PICSEL3), ceux sans radical représentent des primitives du langage RDF. Par exemple, la primitive *type* spécifie le concept dont l'élément considéré est une instance.




```
<p3: Cultural_Object rdf:nodeID="r6o9">
   <p3:name> Château de Chantereine </p3:name>
   <p3:has_Address rdf:nodeID="d1e5" />
</p3: Cultural_Object >

<p3: Postal_Address rdf:nodeID="d1e5" >
   <p3:street> Rue Gauthier </p3:street>
   <p3:zip> 77177 </p3:zip>
   <p3:town> Nemours </p3:town>
   <p3:country> FRA </p3:country>
</p3: Postal_Address >
```

Figures 3.a *Une instance de Cultural_Object* **et 3.b** *sa sérialisation en XML*

Des faits RDF peuvent être représentés de différentes manières en XML. Pour sérialiser le graphe de la figure 3a en RDF-XML, nous retiendrons une écriture compactée qui permet de regrouper les différentes propriétés d'une même instance de concept. Le graphe de la figure 3a représente deux instances de concepts distincts, reliées par la relation *has-Address*. L'écriture présentée en figure 3b conserve cette distinction. Chaque instance est ainsi décrite par un fragment XML délimité par des balises indiquant le type de concept dont elle est l'instance. A chaque instance est associé un identifiant interne. Cet identifiant est repris dans toutes les relations dans lesquelles l'instance intervient. C'est cette référence à l'identifiant qui permet d'effectuer des jointures entre les deux éléments.

3.3. *Les sources XML*

Les sources XML externes sont des sources auxquelles sont associées une DTD (http://www.w3.org/TR/REC-xml/#dt-doctype) et pour lesquelles des mises en correspondance doivent être générées. Les mises en correspondance sont établies entre les DTDs, représentées sous la forme d'arbres, et l'ontologie. La représentation des DTDs est simplifiée. Elle ne prend pas en compte le format des éléments ni des attributs, elle ignore leurs caractéristiques (EMPTY, REQUIRED, etc.) et les cardinalités.

Un arbre T représentant une DTD est un ensemble $T = \{N, A, AT, ATN\}$ où :

- N est l'ensemble fini des nœuds : tous les éléments de cet ensemble sont des éléments de la DTD associée.

- *A* est l'ensemble fini des couples *NxN* représentant des arêtes entre deux nœuds non vides. Ils correspondent aux liens de composition entre éléments de la DTD.
- *AT* est l'ensemble fini des attributs de la DTD. Les attributs de cet ensemble ne sont pas des nœuds de *N* mais chaque attribut est lié à un élément de *N*.
- *ATN* est l'ensemble fini contenant des couples $(e_i , <at_1, at_2, ..., at_n>)$ tel que $i \in \{1,n\}$. Si un élément n'a pas d'attribut, il n'y aura pas de couple de cet élément dans *ATN*. Si un élément e_x a deux attributs at_y et at_z, le couple correspondant dans *ATN* sera $(e_x, <at_y, at_z>)$.

Un exemple de DTD et sa représentation sous forme d'arbre sont donnés figure 4. Dans la suite, nous appellerons *domaine* d'un nœud, l'ensemble de ses nœuds fils dans l'arbre. Ainsi, le domaine du nœud *a* de la DTD de la figure 4 est $\{b,c\}$.

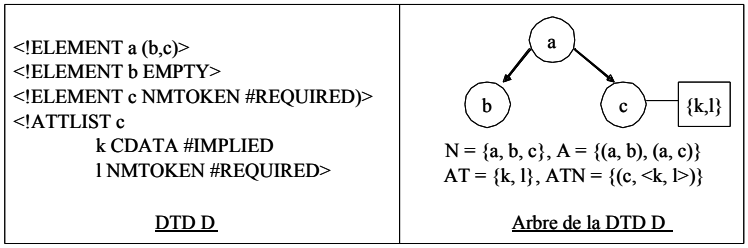


Figure 4. *Une DTD et sa représentation sous forme d'arbre*

A titre d'illustration, la figure 5 présente deux sous-arbres représentant des DTDs. T_1 correspond à une partie de l'arbre de la DTD d'une nouvelle source à intégrer et T_2 à une partie de l'arbre d'une DTD déjà intégrée (la plupart des attributs ne sont pas représentés pour plus de simplicité). Ces DTDs contiennent des nœuds aux noms identiques (en gris dans la figure) : « *poi, name, contact, nb, format, datum, unit, Postal-address* » mais leur position, dans chaque DTD, est différente. Un fragment de document XML extrait de la source à intégrer et satisfaisant le sous-arbre T_1 est présenté figure 6.

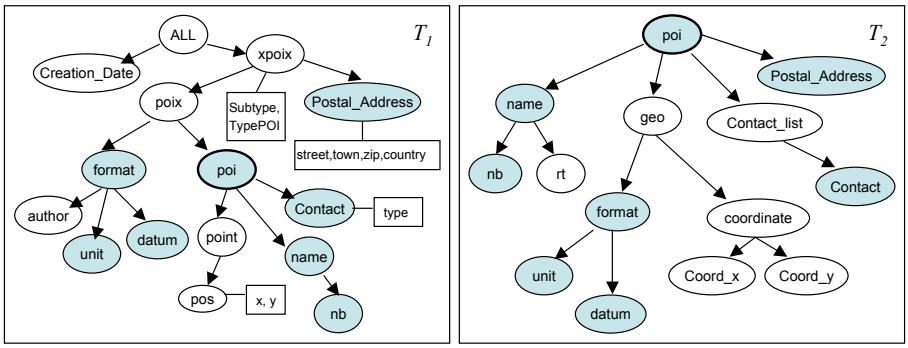


Figure 5. *Sous-arbre T_1 de la DTD à intégrer et sous-arbre T_2 d'une DTD déjà intégrée*

```
<xpoix id = 'PCUIDF07721' typePOI ='touristic_site' subtype = 'castle' >
    <poix version = '1' >
      <format >
          …
      </format >
      <poi>
        <name>
          <nb> Château de Chantereine </nb>
        </name>
        <Contact type = 'tel' > 01 60 20 11 06 </Contact>
        <Contact type = 'fax' > 01 60 20 44 02 </Contact>
        <Contact type = 'email' > marc@chantereine.com </Contact>
      </poi>
    </poix>
    <Postal_Address>
      <town> Nemours </town>
      <street> Rue Gauthier </street>
      <zip> 77177 </zip>
      <country> FRA </country>
    </Postal_Address>
</xpoix>
```

Figure 6. *Fragment de document XML satisfaisant l'arbre T_1*

3.4. *Les mappings*

Un mapping est défini par une fonction qui associe à un concept, une propriété ou une relation de O, l'élément ou l'attribut de la DTD avec lequel ce concept, cette propriété ou cette relation est mis en correspondance, l'élément ou l'attribut étant défini par son chemin dans T, l'arbre de la DTD de la source à intégrer.

Cette mise en correspondance peut être conditionnelle. Les conditions qui doivent être éventuellement satisfaites portent sur les valeurs des éléments ou des attributs dans les documents XML satisfaisant la DTD d'arbre T. Supposons, par exemple, que l'on recherche un mapping pour le concept *email,* spécialisation du concept *contact* de l'ontologie. Un mapping conditionnel le reliera à l'attribut *type* de l'élément *contact* de la DTD, à condition que la valeur de *type* dans le document XML associé à la DTD soit *email*.

Nous utiliserons le symbole "↔" pour représenter la liaison entre les éléments mis en correspondance. Nous explicitons ci-dessous les notations utilisées pour représenter chacune des parties d'un mapping, puis les formats des différents mappings.

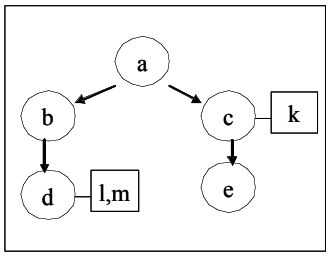
Notation de la composante ontologie. Soient c_1, c_2 deux concepts de l'ontologie O, $r_1 \in R_1$ une relation entre c_1 et c_2 et $r_2 \in R_2$ une propriété de c_1.

- Les concepts de O seront désignés par leur nom, celui-ci les identifiant de manière unique. Les mappings les concernant seront de la forme " c_1 ↔…" et " c_2 ↔…".

- Les relations de R_1 seront désignées par leur nom suivi des concepts liés. Les mappings les concernant seront de la forme "$r_1(c_1, c_2) \leftrightarrow \ldots$".

- Les relations de l'ensemble R_2 (les propriétés) seront désignées par leur nom associé au concept caractérisé. Les mappings les concernant seront de la forme : "r_2 de $c_1 \leftrightarrow \ldots$" et "$r_2$ de $c_2 \leftrightarrow \ldots$".

Notation de la composante source. La deuxième entité liée au sein d'un mapping correspond à un élément ou à un attribut d'un arbre d'une DTD. Chaque élément ou attribut d'un arbre est identifié, de façon unique, par le chemin qui le relie à la racine de l'arbre. Xpath (http://www.w3.org/TR/xpath20/) est utilisé pour représenter ces chemins (cf. figure 7).



/a : désigne la racine de l 'arbre
//c : désigne un élément c à un endroit quelconque du document.
/a/b : désigne un élément b dont le père est a, a étant la racine.
//b/d : désigne un élément d dont le père est b, b n 'étant pas la racine.
//c/@k : désigne l 'attribut k de l 'élément c

Figure 7. *Représentations en Xpath*

Ainsi, les mappings établissant un lien avec un élément dont le nom est *e* seront de la forme : …↔ *//e*. Les mappings établissant un lien avec un attribut (de nom *att*) d'un élément (de nom *e*) de la DTD seront de la forme : ... ↔ *//e/@att*. Les mappings conditionnels établissant un lien avec un attribut d'un élément de la DTD suivant la valeur de cet attribut dans le document XML associé à la DTD, seront de la forme :... *_//e/[@ att='valeur']/@att*. Pour les mappings établissant un lien avec une relation de l'ontologie, la partie du mapping concernant la source désignera le chemin, s'il existe, reliant les correspondants dans la DTD des éléments de l'ontologie reliés par la relation (cf. figure 8).

concept $c_1 \in O$	$c_1 \leftrightarrow$ //e $c_1 \leftrightarrow$ //e/@att $c_1 \leftrightarrow$ //e/[@att = 'val']/@att
relation r_1 entre c_1 et $c_2 \in O$ tels que $\exists$ $c_1 \leftrightarrow$ //a et $c_2 \leftrightarrow$ //b	$r_1(c_1, c_2) \leftrightarrow r_1$(//a, //a/ …/b)
propriété r_2 de $c_1 \in O$ tel que $\exists$ $c_1 \leftrightarrow$ //a et b, correspondant de r_2 dans T	r_2 de $c_1 \leftrightarrow r_2$(//a, //a/ …/b)

Figure 8. *Format des différents mappings*

4. Techniques automatisées de recherche de mappings

L'approche proposée est guidée par l'ontologie. L'objectif est, en effet, d'être capable d'accéder aux données des sources externes et de l'entrepôt à partir d'une requête exprimée dans les termes de l'ontologie. L'approche se décompose en trois étapes. On cherche d'abord à identifier les correspondants des concepts de l'ontologie. On s'appuie ensuite sur les correspondances de concepts trouvées pour chercher les mises en correspondance (ou mappings) entre leurs propriétés. On s'intéresse, dans un troisième temps, aux relations entre concepts. Faute de place, nous ne présentons ici que les techniques permettant d'identifier les correspondants des concepts de l'ontologie. Le processus de découverte de mappings est composé de techniques basées sur les schémas des documents rapprochés (cf. section 4.1) mais aussi sur l'exploitation des données (cf. section 4.2).

4.1. *Exploitation des schémas des documents à rapprocher*

L'objectif est ici de trouver les correspondants des différents concepts *c* de l'ontologie *O* parmi les éléments d'une DTD d'arbre T_1. Deux techniques sont utilisées, l'une exploitant l'ontologie, la seconde exploitant les mappings pré-existants, la première précédant la seconde dans son application.

La première technique appliquée consiste à poser que si, pour un concept *c* de l'ontologie *O*, il existe dans T_1 un élément de même nom que *c*, un mapping peut être établi entre le concept et l'élément considéré. Cette technique est justifiée par le fait que la DTD est construite à l'aide du vocabulaire fixé dans l'ontologie *O*, que l'ontologie, telle qu'elle est structurée, correspond à une description des connaissances reflétant un certain point de vue, que le concepteur de la DTD connaît ce point de vue et choisit, en conséquence, les termes de l'ontologie pour structurer son document. Si un même terme est utilisé dans l'ontologie et le schéma d'une source, c'est que le concepteur de la nouvelle source est d'accord avec l'interprétation proposée par le concepteur de l'ontologie.

La deuxième technique s'applique quand la première a échoué, c'est-à-dire quand il n'existe pas de termes dans T_1 de même nom qu'un concept *c* de l'ontologie. Elle consiste à utiliser la liste des mappings pré-existants entre *O* et les DTDs des sources déjà intégrées dans l'entrepôt local. On recherche à quel terme le concept *c* a été connecté dans les autres DTDs. Si pour une DTD d'arbre T_2, un mapping existe pour le concept *c* avec un élément *e* présent également dans T_1, l'approche consiste à étudier si ce mapping ne pourrait pas aussi s'appliquer dans le contexte de T_1. Le mapping n'est pas immédiatement réutilisé car il n'est pas certain que les deux termes de même nom mais utilisés dans des contextes différents correspondent au même concept. Une vérification s'impose. Elle est basée sur la comparaison du domaine (D_2) du terme considéré dans l'arbre T_2 avec le domaine de ce même terme ou de ses ancêtres dans l'arbre T_1. Une telle comparaison permet de trouver l'élément *e*' de T_1 le plus proche du *e* de T_2, c'est-à-dire dont le domaine a le

plus d'éléments en commun avec le domaine D_2 de e dans T_2. L'élément e' retourné est soit l'élément e initial, soit un ancêtre de e dans T_1.

La comparaison s'effectue de la façon suivante. Au départ, le nombre de termes communs aux deux domaines comparés, N_{inter}, est initialisé à 0. En cours d'exécution, si N_{inter} est égal au nombre d'éléments de D_2, cela signifie que les deux domaines comparés sont égaux, ou que le domaine D_2 de e dans T_2 est inclus dans le domaine de l'élément comparé dans T_1. Nous considérons, dans ce cas, que l'élément le plus proche du e de T_2 est l'élément avec lequel il est comparé dans T_1. Il pourrait être possible de trouver un spécialisant de l'élément comparé dans T_1 qui satisfasse aussi cette condition portant sur les domaines. Notre approche ne le recherche pas dans la mesure où le premier élément trouvé permet d'établir un mapping qui, lorsqu'il sera utilisé, permettra de renvoyer une réponse englobant toutes les données attendues. Dans le cas contraire, si le nombre d'éléments communs lors d'une étape est supérieur à celui de l'étape précédente, le processus est réitéré avec le père de l'élément courant dans T_1. Lorsque le nombre d'éléments communs ne progresse plus, l'élément recherché retourné est l'élément courant de l'étape précédente.

A titre d'illustration, considérons les arbres de DTD T_1 et T_2 présentés en figure 5 et suivons l'exécution de l'algorithme pour *poi* (abréviation de point d'intérêt), nœud commun aux deux DTDs.

Recherche du correspondant d'un concept c de l'ontologie parmi les éléments d'une DTD $T_1 = \{N_1, A_1, AT_1, ATN_1\}$

Si $c \in N_1$ Alors retourner $c \leftrightarrow$ //c
Sinon
  Si $\exists$ $T_2 = \{N_2, A_2, AT_2, ATN_2\}$ déjà intégré et
    $\exists$ e, tel que $e \in N_2$
      et $e \in N_1$
      et $\{c \leftrightarrow$ //e $\} \in$ ListeMappingEtablis
  Alors $D_2 \leftarrow$ Domaine(e,T_2) ;
    e' $\leftarrow$ Cdom(e, T_1, D_2, 0, e) ;
    retourner $c \leftrightarrow$ //e'
  FinSi
FinSi

Cdom(e, T_1, D_2, Seuil, Origine)
  $D_1 \leftarrow$ Domaine(e,T_1) ; $N_{inter} \leftarrow D_1 \cap D_2$;
  Selon le cas N_{inter}
    $N_{inter} = |D_2|$: retourner e ;
    $N_{inter} >$ Seuil : Cdom(père(e), T_1, D_2, N_{inter}, e) ;
    $N_{inter} =$ Seuil : retourner Origine
  FinSelon

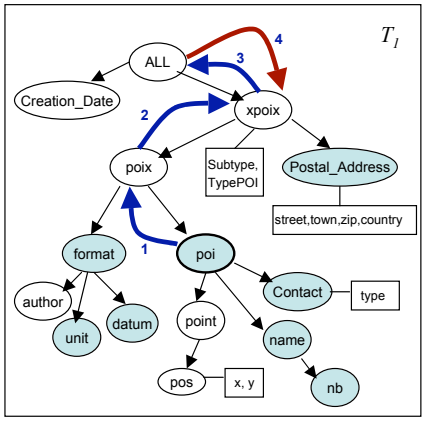


Figure 9. *Illustration du déroulement de l'algorithme comparant les domaines*

Le domaine D_2 de *poi* dans T_2, {*name, nb, rt, geo, format, unit, datum, coordinate, coord_x, coord_y, contact_list, contact, Postal_address*}, de cardinalité 13, est comparé au domaine de l'élément courant *poi* dans T_1, {*point, pos, name, nb, contact*}. Trois éléments, {*name, nb, contact*}, sont communs aux deux domaines. N_{inter} étant différent du cardinal de D_2 mais supérieur à N_{inter} initialisé à 0, le processus est réitéré en comparant D_2 au domaine du nœud père de *poi* dans T_1, l'élément *poix* (flèche numéro 1 sur la figure 8). Six éléments sont communs :

{*name, nb, contact, format, unit, datum*}. N_{inter} ayant augmenté, le processus est réitéré à partir du père de poix, *xpoix* (flèche 2) puis à partir du père de *xpoix*, *ALL* (flèche 3). A ce stade, le nombre d'éléments communs n'augmente plus, l'élément de T_1 dont le domaine est le plus proche de celui de *poi* dans T_2 est donc *xpoix*, l'élément courant de l'étape précédente (flèche 4). De cette façon, le concept *Cultural_Object* de *O* qui n'a pas de correspondant dans T_1 mais pour lequel un mapping a déjà été établi avec le terme *poi* dans T_2 sera mis en correspondance avec *xpoix* dans T_1.

4.2. *Exploitation des données des documents à rapprocher*

Les techniques considérées dans cette section exploitent les données contenues dans les documents XML. Cette étude des données s'explique par le fait que les concepteurs des sources XML n'utilisent pas toujours des noms de balise aussi précis que ceux proposés dans l'ontologie. Une pratique relativement courante consiste à définir des éléments relativement généraux puis à les caractériser en fixant les valeurs de certains de leurs attributs. En revanche, dans l'ontologie, si des concepts ont des spécificités, ils sont généralement modélisés sous forme de concepts liés par une relation de spécialisation à des concepts plus généraux. Par exemple, dans l'ontologie *OntoTourism* avec laquelle nous avons travaillé, plusieurs concepts (*email, fax, tel, web)* ont été définis comme des spécialisations du concept *Contact*. Ces noms de concepts sont utilisés par certains concepteurs de sources comme des noms de balises explicites, mais d'autres concepteurs peuvent préférer représenter les spécificités des contacts via la valeur d'un attribut (en général *Type* ou *Subtype*). Ainsi, un email pourra être représenté comme un *Contact* dont la valeur de l'attribut *Type* est *email*.

Ce choix de représentation a un impact sur les mappings à générer car les noms de concept ou de propriété qui n'interviennent pas explicitement en tant que balise n'apparaissent pas dans les arbres de DTD. Cet état de fait est pris en compte dans *DétectHybrideMap* par les mappings conditionnels, qui mettent en correspondance un concept de l'ontologie et un élément d'une DTD lorsqu'un de ses attributs prend une valeur donnée. Ainsi par exemple, le mapping concernant *email*, est représenté de la façon suivante : *email* ↔ //*Contact*[@*type* ='*email*']/@*type*, ce qui correspond à mettre en correspondance *email* de l'ontologie avec *contact* de la source à intégrer à condition que la valeur de l'attribut *type* de *Contact* soit *email*.

La recherche d'éventuels mappings conditionnels avec l'arbre T_1 d'une nouvelle source à intégrer s'appuie sur les mappings conditionnels déjà établis. L'identification de nouveaux mappings de ce type impose de vérifier qu'il existe dans T_1 un élément dont les valeurs dans les documents XML sont celles exprimées dans les conditions des mappings conditionnels déjà établis. Cette vérification sera faite parmi les termes de T_1 pour lesquels aucun mapping n'a encore été établi.

Pour des raisons d'efficacité, l'approche proposée pour effectuer la vérification consiste à partitionner l'ensemble des mappings conditionnels déjà établis en

différentes classes selon l'élément avec lequel la mise en correspondance est établie. On obtiendra ainsi, par exemple, la classe des mappings conditionnels établis avec *Contact*. A chaque classe de mappings sera associé un ensemble de valeurs, celles intervenant dans les conditions des mappings. {*tel, fax, email, web*} sera l'ensemble de valeurs associées à la classe *Contact*. La construction de ces ensembles de valeurs nécessite d'exploiter les données des documents XML proprement dites. Cherchant à minimiser le volume des données à exploiter, seul un sous-ensemble des documents de l'entrepôt local sera considéré. La taille du sous-ensemble optimal à considérer a été déterminée de manière empirique par expérimentation (cf. section 6).

Cette troisième technique consiste ainsi à travailler sur chaque concept *c* de l'ontologie qui n'a pas encore de correspondant dans T_1 mais pour lequel existe, dans la liste des mappings déjà établis, un mapping conditionnel avec un élément d'une DTD d'arbre T_2. Soit la classe de mapping Cl_i à laquelle appartient le mapping considéré et son ensemble de valeurs V_i, l'élément de T_1 retenu pour le mapping sera celui dont le score, rapport entre le nombre de valeurs appartenant à V_i prises par un attribut de cet élément et le cardinal de V_i, sera le plus élevé. Une proposition de mapping est faite à l'expert pour tous les éléments de T_1 dont le score est non nul. Par exemple, soit le concept *Museum* de *O* qui n'a pas encore de correspondant dans T_1 mais pour lequel a déjà été établi le mapping conditionnel suivant dans T_2 : Museum ↔ //category[@subtype='museum']/@subtype. Ce mapping appartient à une classe dont l'ensemble des valeurs est {*museum, touristic-site*}. L'attribut *typePOI* de l'élément *xpoix* de T_1 ayant aussi pour liste de valeurs l'ensemble {*museum, touristic-site*}, le score de *xpoix* est 2/2, et les deux mappings conditionnels ci-dessous sont proposés à l'expert pour validation :

Museum ↔ //*xpoix*[@*typePOI* ='*museum*']/@*typePOI*
Touristic-site ↔ //*xpoix*[@*typePOI* ='*touristic-site*']/@*typePOI*

D'autres techniques d'exploitation des données des documents sont nécessaires. Un premier travail a été initié. Il consiste à rapprocher des éléments de noms syntaxiquement différents, qui n'apparaissent pas dans des mappings conditionnels, mais dont les données associées ont le même format (date, monétaire, entier, réel, alpha-numérique, booléen). Une telle technique peut être intéressante pour des concepts ou des attributs ayant un format particulier, par exemple le format date, monétaire, etc. Ainsi, la propriété *coord_1* du concept *Geo* de l'ontologie représentant un lieu géographique, n'a pas de correspondant dans l'arbre T_1 de la nouvelle source à intégrer. Il existe pour cette propriété un mapping avec l'élément *coord_x* de T_2. L'élément *coord_x* n'existe pas dans T_1 mais il a le même format de valeurs que l'attribut *x* de l'élément *pos* de T_1 sans correspondant dans l'ontologie. Selon cette technique, un mapping entre *coord_1* et *x* pourrait être proposé à l'expert pour validation. Dans l'état actuel de l'implémentation, cette proposition n'est pas faite car deux candidats au mapping sont possibles : *x et y,* et le choix a été fait de ne faire aucune proposition en cas de mappings possibles multiples.

5. Construction automatique de wrappers

La construction d'adaptateurs, ou wrappers, peut être décomposée en deux phases. Une première phase construit une description sémantique et abstraite du contenu de la source, dont les données doivent être extraites, exprimée sous la forme de vues dans le langage de la plate-forme. Une deuxième phase construit la partie effective du wrapper associée à chaque vue, i.e. la requête qui permet d'extraire les données décrites dans la vue et de les représenter directement dans le langage de l'entrepôt.

Pour des raisons d'efficacité, les constructions des vues et des requêtes sont réalisées dans le même processus. Mais pour des raisons de lisibilité, nous commencerons par décrire le mécanisme de construction des vues, avant de présenter la génération des requêtes. Nous présentons tout d'abord le format des vues construites, la stratégie générale de construction adoptée puis la description détaillée du processus.

5.1. *Type de vues construites*

Une vue est une implication logique reliant un nom local de vue, v_i, à un concept ou à une relation du domaine, $p(x)$, dont on sait que la source peut fournir des instances, $v_i(x) \rightarrow p(x)$. Ainsi le contenu sémantique d'une source externe peut être représenté, en intension, par autant de vues que de concepts ou de relations du domaine dont on sait que la source peut fournir des instances. Par exemple, si une source S_1 contient des instances de lieux culturels avec leur nom et leur adresse, la description de cette source pourrait être représentée par quatre vues distinctes, explicitant le fait que la source permet d'obtenir à la fois des instances du concept *Cultural-Object*, des instances de *name* de lieu culturel, des instances du concept *Postal_Address*, et des instances de relations *has-Address* reliant les adresses aux lieux culturels concernés.

Dans l'approche suivie dans ce travail, au lieu de construire une vue distincte pour chaque élément du domaine représentable, nous avons préféré construire un nombre plus limité de vues, en associant directement au sein d'une même vue un concept, dit *concept central* de la vue, et les propriétés et les relations qui lui sont rattachées. Ainsi la source S_1 permettant de fournir des instances de *Cultural-Object* comme celles présentées en figures 3a et 3b sera décrite par une unique vue, regroupant à la fois le concept *Cultural-Object* et ses relations associées :

$$S_1(x,y,a,s,z,t,c) \rightarrow \textit{Cultural-Object}(x) \wedge \textit{name}(x,y) \wedge \textit{has-Address}(x,a) \wedge \textit{Postal\_Address}(a) \wedge \textit{street}(a,s) \wedge \textit{zip}(a,z) \wedge \textit{town}(a,t) \wedge \textit{country}(a,c)$$

Cette vue de S_1 fournit, en même temps, des instances de deux concepts distincts, des instances de *Cultural-Object*, le concept central de la vue, et des instances de *Postal_Address*, concept atteint par une relation associée au concept central. Ce

dernier concept sera dit *concept subordonné*. Si un concept *subordonné* possède lui-même dans la source des propriétés et des relations, ces propriétés et relations sont elles aussi récursivement introduites dans la vue décrivant le concept central.

L'intérêt de ce regroupement par rapport à des vues éclatées est de faciliter par la suite l'accès aux propriétés ou relations d'un concept, en évitant de coûteuses opérations de jointure. Nous verrons aussi plus loin, qu'une fois les données extraites de la source et transférées dans l'entrepôt, l'accès direct aux instances d'un concept subordonné quelconque ne posera aucune difficulté.

Ainsi, la description d'une source sera composée d'autant de vues que de concepts, non subordonnés à un autre concept, décrits dans la source.

5.2. *Stratégie de construction d'une vue*

Chaque vue relative à un concept est construite à partir de ses propriétés, de ses relations et des concepts liés. La construction d'une vue est incrémentale et guidée, dans un premier temps, par la DTD de la source, puis dans un deuxième temps par l'ontologie du domaine. La partie du processus de construction guidée par la DTD permet d'identifier les concepts centraux pour lesquels une vue doit être construite. La partie guidée par l'ontologie permet l'identification des propriétés et des relations associées au concept à introduire dans la vue.

La partie guidée par la DTD consiste à faire un parcours en profondeur (préfixé) de cette DTD, élément par élément (depuis la racine), jusqu'à rencontrer le premier élément de la DTD, e_D, pour lequel on dispose d'un mapping avec un élément e_O de l'ontologie. Dans notre contexte, du fait de la définition des mappings, l'élément e_O de l'ontologie associé par le mapping à l'élément e_D ne peut être qu'un concept et ce concept ne peut pas être subordonné à un autre concept de la DTD. C'est donc un concept central, pour lequel une vue doit être construite par la partie du processus guidée par l'ontologie. La recherche d'éventuels autres concepts non subordonnés reprendra en poursuivant l'exploration de la DTD en partant du frère de l'élément e_D considéré. En effet, a priori, tous les éléments apparaissant dans le sous-arbre de la DTD de racine e_D et disposant de mappings devraient être reliés dans l'ontologie à l'élément e_O.

EXEMPLE. — Prenons la partie de la DTD représentée sur la figure 10, extraite de la figure 5. Le nœud racine, *ALL,* n'a pas de correspondant dans *O*. Le parcours en profondeur passe alors à l'étude de son premier fils, *Creation_Date*, qui n'a pas non plus de correspondant dans *O* et n'a pas de fils. C'est le deuxième fils de *ALL*, *xpoix*, qui a pour correspondant dans *O* le concept *Cultural-Object*, qui est alors étudié et qui correspondra au premier concept central identifié. Une vue associée à l'élément correspondant de la DTD est créée : $S_1(x) \rightarrow Cultural\text{-}Object(x) \wedge \ldots$ Les éventuels frères de *xpoix* seront traités ultérieurement.

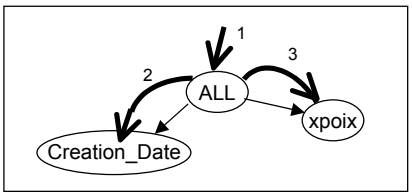


Figure 10. *Exemple de parcours d'une DTD*

Etant donné un concept de l'ontologie identifié dans le parcours de la DTD comme étant un concept central d'une vue, la partie du processus guidée par l'ontologie consiste à rechercher dans l'ontologie les propriétés et les relations associées à ce concept et à introduire dans la vue toutes celles qui disposent de mappings dans la DTD (et uniquement celles-ci). Le processus est effectué récursivement sur tous les concepts subordonnés en relation avec le concept considéré.

5.3. *Description détaillée du processus*

La construction de la vue associée à un concept central, dite encore *vue globale*, est incrémentale. Elle se fait au fur et à mesure que l'on extrait les entités reliées au concept central et de façon *récursive* sur les concepts subordonnés. En exploitant l'ontologie, pour chaque concept introduit dans une vue, nous traiterons successivement ses propriétés, ses relations associées, ses spécialisations, en n'introduisant dans la vue que les éléments disposant de mappings.

5.3.1. *Traitement des propriétés d'un concept*

Pour chaque prédicat représentant dans l'ontologie une propriété associée au concept considéré, on introduit, dans le nom local de la vue, un nouvel argument sous la forme d'une nouvelle variable, et dans la partie droite de la vue, le nom du prédicat de la propriété. Ce prédicat a deux arguments, le premier est la variable associée au concept en cours de traitement, le deuxième, la nouvelle variable introduite.

EXEMPLE. — Dans l'ontologie, le concept *Cultural-Object* possède trois propriétés, *name*, *keyword* et *summary*, mais seule la propriété *name* dispose d'un mapping dans la DTD. Ce sera donc le seul nouveau prédicat introduit dans la vue à cette étape : $S_1(x,y) \rightarrow Cultural\text{-}Object(x) \wedge name(x,y)$…

5.3.2. *Traitement des relations associées à un concept*

Pour chaque prédicat représentant dans l'ontologie une relation associée au concept considéré, on introduit dans le nom local de la vue, une nouvelle variable en argument. Dans la partie droite de la vue, on introduit à la fois :

- le nom du prédicat de la relation avec deux arguments, le premier étant la variable associée au concept en cours de traitement, le deuxième, la nouvelle variable introduite,
- le nom du prédicat représentant le concept relié par la relation (concept subordonné au concept en cours de traitement), avec comme argument la nouvelle variable.

EXEMPLE. — Deux relations associées au concept *Cultural-Object* disposent de mappings, la relation *has_Address* qui relie *Cultural-Object* à son adresse et la relation *has_Contacts* qui le relie au concept *Contact*. Après le traitement de ces deux relations, la vue en cours de construction devient *: $S_1(x,y,z,t) \rightarrow$ Cultural-Object(x) ∧ name(x,y) ∧ has_Address(x,z) ∧ Postal_Address(z) ∧ has_Contacts(x,t) ∧ Contact(t)…*

Chacun des concepts subordonnés introduits dans la vue au cours du traitement d'un concept est à son tour traité récursivement et ses prédicats associés (propriétés, relations et concepts subordonnés) sont introduits dans la vue.

5.3.3. *Traitement des spécialisations d'un concept*

Deux types de traitements distincts ont été définis suivant que le concept qui peut être spécialisé est un concept central ou subordonné, et, pour les concepts subordonnés, suivant la cardinalité de la relation qui les lie.

Si le concept possédant des spécialisations dans l'ontologie est un concept central, on créera autant de copies distinctes de la vue que de spécialisations du concept, en remplaçant dans chaque copie le nom du concept par le nom de sa spécialisation et en modifiant le nom local de la vue.

EXEMPLE. — Si le concept central de la vue, *Cultural-Object*, a deux spécialisations, *Castle* et *Museum*, on créera deux copies de la vue complète, $S_1(x,y,z,t)$.

D'une part : $S_2(x,y,z,t) \rightarrow$ *Castle(x) ∧ name(x,y) ∧ has_Address(x,z) ∧ Postal_Address(z) ∧ has_Contacts(x,t) ∧ Contact(t).*

D'autre part : $S_3(x,y,z,t) \rightarrow$ *Museum(x) ∧ name(x,y) ∧ has_Address(x,z) ∧ Postal_Address(z) ∧ has_Contacts(x,t) ∧ Contact(t).*

Si le concept possédant des spécialisations dans l'ontologie est un concept subordonné, le traitement dépend de la cardinalité de la relation qui établit un lien avec ce concept. Par exemple, le concept *Contact*, qui a pour spécialisations les concepts *Tel, Fax, Email* et *Web*, a été introduit dans la vue par la relation *has_Contacts* liée au concept *Cultural-Object*. *has_Contacts* permet d'associer plusieurs contacts distincts à un même lieu culturel. Dans ce cas, pour chaque prédicat représentant dans l'ontologie une spécialisation du concept *Contact*, on introduit dans la vue, à la fois, un nouvel argument sous la forme d'une nouvelle variable, une nouvelle occurrence de la relation *has_Contacts(x,t)* ainsi que le prédicat représentant la spécialisation.

EXEMPLE. — $S_1(x,y,z,t,t_1,t_2,t_3,t_4) \rightarrow$ *Cultural-Object(x) ∧ name(x,y) ∧ has_Address (x,z) ∧ Postal_Address(z) ∧ has_Contacts(x,t) ∧ Contact(t) ∧ has_Contacts(x,* t_1*) ∧*

$Tel(t_1) \wedge has\_Contacts(x, t_2) \wedge Fax(t_2) \wedge has\_Contacts(x, t_3) \wedge Email(t_3) \wedge has\_Contacts(x,t_4) \wedge Web(t_4)$ …

Si la cardinalité de la relation qui établit un lien avec le concept subordonné ne permet de mettre en relation qu'au plus un unique élément, le traitement des spécialisations sera le même que dans le cas du concept central, i.e. on créera autant de copies de vues distinctes que de spécialisations du concept.

Pour des raisons d'efficacité, le traitement des spécialisations de concept menant à créer des copies de vues est effectué en dernier, de façon à ne créer que des copies de vues complètes, i.e. dans lesquelles tous les prédicats nécessaires ont été introduits.

5.4. *Extraction des données*

La construction des vues présentées précédemment permet d'obtenir une définition en intension du contenu d'une source, formée d'autant de vues que de concepts centraux dont la source peut fournir des instances. Pour chaque vue, l'extraction effective de ses instances passe par la réalisation d'un adaptateur qui permet d'interroger la source dans son langage spécifique et de transposer les instances dans le formalisme de représentation choisi pour cible, ici RDF, le langage de l'entrepôt, en adoptant la syntaxe RDF/XML présentée en section 3.2.

Les wrappers sont réalisés à l'aide de requêtes XQuery (http://www.w3.org/TR/xquery). XQuery est le langage de requête recommandé par le W3C permettant d'interroger de données XML. Chaque wrapper associé à une vue sera représenté par une requête en XQuery. Pour cela, nous utiliserons des expressions *FLWOR* où *FLWOR* est l'acronyme de *F*or, *L*et, *W*here, *O*rder by, *R*eturn. La syntaxe d'une expression *FLWOR* est la suivante :

FLWOR Expression := (ForClause | LetClause) + WhereClause ? + OrderByClause ? + "ReturnClause".

La clause *For* (de la forme For $x in $doc//element) considère chaque élement de *element* du document $doc. La clause *Let* déclare une variable et lui affecte une valeur. La clause *Where* permet de spécifier un critère de sélection. La clause *Return* spécifie l'expression à retourner. Ainsi dans une expression FLWOR, la partie FLWO correspond à la composante extraction du wrapper et la partie R (pour *Return*) à la composante génération du résultat au format RDF/XML. Cette dernière composante est réalisée en utilisant les termes de l'ontologie comme noms de balises délimitant les résultats de la requête XQuery.

La construction des wrappers suit le processus de construction des vues. La construction de la requête qui représentera le wrapper associé à un concept central, est effectuée de façon incrémentale en traitant le concept, puis ses propriétés, puis les concepts qui lui sont subordonnés. Le processus se termine par le traitement des spécialisations des concepts centraux.

5.4.1. *Initialisation d'une requête associée à un concept central*

La construction de la requête s'appuie sur les mappings identifiés entre les termes de l'ontologie et ceux de la DTD. Pour un concept central nommé *conceptC* dans l'ontologie, on identifie son correspondant *mapC* au sein de la liste des mappings fournie en entrée. C'est ce correspondant *mapC* dont on recherchera les instances dans la clause *FOR*.

Chaque instance de concept devant être identifiée par un identificateur unique, la génération automatique d'identificateurs au sein d'une requête XQuery se fait à l'aide d'une fonction spécifique java (*generate-Id*) qui sera exécutée en même temps que la requête. L'identifiant généré sera introduit sous la forme d'un *rdf :nodeID*. Le nom du concept dans l'ontologie est utilisé comme nom de balise dans la partie *Return*. La forme initiale de la requête est donc la suivante :

```
for $x in doc("source.xml")//mapC
  let $idcpt := gi:generate-Id()
return
<p3:conceptC rdf:nodeID="{$idcpt}">
```

Le fragment de données désigné par *$x* est associé au concept central (dans l'exemple, *//mapc*). Ainsi, pour construire la requête associée à la vue S1(x) → *Cultural-Object*(x) ∧ …, le correspondant du concept central *Cultural-Object* étant l'élément *xpoix*, le début de requête est : **for** $x in doc("source.xml")//*xpoix* (cf. ligne 4 de la requête ci-dessous).

Le fragment désigné par $x comprend l'ensemble des éléments qui le composent dans le document XML. La requête XQuery consistera alors à extraire de ce fragment l'ensemble des propriétés et relations reconnues dans l'ontologie pour ce concept, i.e. celles pour lesquelles il existe un mapping dans l'ontologie.

5.4.2. *Extraction des propriétés d'un concept*

Le correspondant dans la DTD d'une propriété *prop* d'un concept est représenté par un chemin *mapProp* issu du correspondant du concept (cf. section 3.4). L'extraction de la propriété en XQuery se fait donc en utilisant ce chemin et la primitive *text()* pour extraire l'élément sans ses balises : <p3:*prop*>{$x/*mapProp*/text()}</p3:*prop*>.

EXEMPLE. — Soit la propriété *name* de *Cultural-Object,* dont le mapping est *name* de *Cultural-Object* ↔ *name*(//*xpoix*, *//xpoix/poix/poi/name/nb*), la partie de la requête associée est : <p3:*name*>{$x/*poix/poi/name/nb*/text()}<p3:*name*>. Nous remarquons que *xpoix*, racine du chemin XPath, n'est pas repris car cet élément est référencé par $x dans la requête.

5.4.3. *Extraction des relations associées à un concept*

Pour matérialiser la relation rel_i entre le concept *conceptC* et son concept subordonné *conceptS*, et pour expliciter leur jointure en RDF, nous générons un nouvel identificateur pour le concept subordonné : let $idrel$_i$:= gi:generate-Id(). La relation et la référence à cet identificateur sont explicitées dans le fragment XML décrivant *conceptC* (<p3:rel_i rdf:nodeID="{$idrel$_i$ }"/>) et un nouveau fragment est construit pour décrire le concept subordonné et ses propriétés de la forme *mapPropS*) :

```
<p3:conceptS rdf:nodeID="{$idrel_i}">
  <p3:propS>{$x/mapRel_i/mapPropS/text( )}</p3:propS>
</p3:conceptS >
```

La requête ci-dessous permet d'extraire les données associées à la vue S_1 telle que $S_1(x,y,z,t) \rightarrow Cultural\text{-}Object\ (x) \wedge name(x,y) \wedge has\text{-}Address(x,z) \wedge Postal\_Address(z) \wedge Town(z,t)$:

```
1. declare namespace gi = "java:pkg.GenerateId";
2. declare namespace rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#";
3. declare namespace p3 ="http://www.lri.fr/picsel3/tourismrdfs#";
4. for $x in doc("source.xml")//xpoix
5.     let $idcpt := gi:generate_Id()
6.     let $idrel := gi:generate_Id()
7. return
8. <p3: Cultural_Object rdf:nodeID="{$idcpt}" >
9.     <p3:name> {$x/poix/poi/name/nb/text()} </p3:name>
10.    <p3:has_Address rdf:nodeID="{$idrel}"/>
11. </p3:Cultural_Object>
12. <p3: Postal_Address rdf:nodeID="{$idrel}">
13.    <p3:town> {$x/Postal_Address/town/text()}</p3:town>
14. </p3: Postal_Address>
```

5.4.4. *Traitement des spécialisations d'un concept*

Une fois les spécialisations explicitées dans les vues, leur traitement ne pose pas de problèmes particuliers. Pour les concepts dont les différentes spécialisations doivent être introduites dans la même vue, on créera un identificateur pour chaque spécialisation du concept et on l'introduira dans la requête avec une instance de la relation.

Pour les concepts menant à la création de copies de la vue, on créera une copie de la requête en remplaçant, dans les balises, le nom du concept par le nom de la spécialisation.

5.4.5. *Traitement des mappings conditionnels*

Nous avions vu en section 3.4 que certains mappings peuvent être conditionnels, i.e. n'être effectifs que si une certaine condition est remplie. Par exemple, le concept *Castle* qui est une spécialisation du concept *Cultural_Object*, a pour correspondant *xpoix* si l'attribut *subtype* de *xpoix* a la valeur *castle* : $Castle \leftrightarrow$ //*xpoix*[@*subtype*='*castle*']/@*subtype*. La partie conditionnelle du mapping sera introduite dans la requête sous la forme d'une clause *Where*. Ainsi la requête associée à la vue S_{11} telle que $S_{11}(x, y) \rightarrow Castle(x) \wedge name(x,y)$ est :

```
for $x in doc("source.xml")//xpoix
  let $idcpt := gi:generate-Id()
where $x/@subtype="castle"
return
<p3:Castle rdf:nodeID="{$idcpt} ">
    <p3:name>{$x/poix/poi/name/nb/text()} <p3:name>
</p3:Castle>
```

6. Expérimentations

Les expérimentations de *DétectHybrideMap* ont été réalisées à partir de données réelles, relatives à des sites culturels, fournies par France Télécom Recherche & Développement (FT) et par le Comité Régional du Tourisme de l'Île-de-France (CRT). L'ontologie *OntoTourism* relative aux données étudiées comprend 34 concepts, 27 relations (8 sont des relations entre concepts, 19 sont des propriétés de concepts).

6.1. *Génération des DTDs et des arbres associés*

Les données nous ont été transmises sous forme de documents XML. Les données du CRT, à l'origine représentées dans un format propre, ont été traduites par FT qui y a introduit des balises identiques pour la plupart à celles utilisées dans ses propres documents. Elles se présentent sous la forme d'un fichier XML qui décrit 730 sites culturels de Paris, tous structurés de la même façon et encadrés par une balise, nommée *ALL*. Une DTD a été générée à partir de ce fichier. Les données propres à FT se présentent sous la forme de 536 fichiers XML, tous structurés de la même façon et décrivant chacun un site culturel de Paris. Une DTD compatible avec l'ensemble de ces fichiers a été générée. Les DTDs générées ont été simplifiées afin d'obtenir des arbres ne comprenant que les noms des éléments et des attributs sous forme de balise. Il s'agit d'une copie exacte des DTDs des sources sans cardinalité et sans indication des types.

6.2. *Etablissement des mappings*

Nous avons établi manuellement des mappings entre l'ontologie *OntoTourism* et, d'une part, l'arbre de DTD des données propres à FT, et d'autre part, l'arbre de DTD des données du CRT. La première liste de mappings (pour les données FT) représente l'ensemble de mappings supposés déjà établis dans notre approche, et la deuxième liste, (pour les données du CRT), ceux que notre système doit être capable de générer automatiquement, à partir de l'arbre de DTD de la nouvelle source. 58 mappings ont été établis entre l'ontologie et l'entrepôt local (couvrant 95 % des termes de l'ontologie) et 51 mappings avec la nouvelle source à intégrer (couvrant 83 % des termes).

6.3. *Tests réalisés*

L'Algorithme *DétectHybrideMap* a réussi à détecter 90,1 % des mappings entre les termes de *OntoTourism* et ceux de CRT. La distribution des mappings détectés suivant la technique utilisée par *DétectHybrideMap* est donnée dans Table 1. Les catégories 1 et 2 correspondent aux mappings portant sur des concepts ou des propriétés de l'ontologie identifiés par exploitation du schéma (cf. section 4.1). Plus précisément, la catégorie 1 regroupe les mappings entre deux entités ayant le même nom. La catégorie 2 regroupe des mappings entre deux entités dont le nom est différent mais dont une mise en correspondance a déjà été établie sur une autre source. Les catégories 3 et 4 correspondent aux mappings identifiés par exploitation des données des sources (section 4.2). La catégorie 3 regroupe des mappings conditionnels entre un concept de l'ontologie et une valeur d'un attribut ou d'un élément de la DTD. La catégorie 4 regroupe des mappings entre un concept ou une propriété de l'ontologie et un élément (au sens général du terme) de la DTD dans le cas où aucune correspondance n'a pu être établie entre les noms des éléments ou en s'appuyant sur des mappings existants dans d'autres sources. Le travail concernant les mappings de cette catégorie a été juste initié et il y est fait référence à la fin de la section 4.2. La dernière technique (non présentée dans le papier) établit un mapping pour une relation de l'ontologie quand celle-ci relie deux concepts de l'ontologie pour lesquels des mappings ont été identifiés. Le mapping consiste à associer la relation au chemin reliant les correspondants des deux concepts dans l'arbre s'il existe (cf. section 3.4).

Pour les mappings conditionnels, nous avons fait plusieurs essais afin de trouver la taille minimale de l'échantillon à exploiter pour qu'il soit représentatif. Nous avons d'abord calculé les scores de chaque classe de mappings (cf. section 4.2) en exploitant tous les documents XML. Le score d'un élément donné d'une DTD pour une classe de mappings conditionnels est le rapport entre le nombre de valeurs de la classe prises par cet élément et le nombre de valeurs de la classe. Nous avons ensuite réduit le nombre des documents utilisés par pas de cinq pour cent. Nous avons constaté qu'au-dessous de 25%, les scores de chaque classe de mappings

diminuaient beaucoup. La taille de l'échantillon a donc été fixée à 25% de l'ensemble des documents.

Les mappings conditionnels détectés sont corrects. Néanmoins, le système considère qu'il ne s'agit que de propositions. Il les présente, accompagnés de leur score, au concepteur du système qui doit les confirmer ou les refuser.

Trois mappings dépendant de la technique de comparaison du format des valeurs (catégorie 4) n'ont pas été trouvés par le système. Il s'agit du mapping rapprochant les concepts *appreciation* de l'ontologie et *quality* de la DTD de CRT, et ceux rapprochant les propriétés *coord_1* et *coord_2* du concept *Geo*, des attributs *x* et *y* de l'élément *pos* de la DTD de CRT. Pour détecter les mappings de ce type, il sera indispensable de faire une analyse plus profonde, comme une analyse sémantique afin d'améliorer le taux de détection des mappings.

Les catégories	1	2	3	4	5	Total des mappings
Mappings de France Telecom	19	2	25	7	5	58
Mappings de CRT	16	2	25	5	3	51

Les catégories	1	2	3	4	5	Total des mappings
Mappings de CRT établis manuellement	16	2	25	3	5	51
Mappings détectés	15	2	25*	0	4	46

Table 1. – *Répartition des mappings manuels et détectés par catégorie.*

Les deux mappings de relation qui dépendaient de l'existence d'un mapping pour le concept *appreciation* n'ont de ce fait pas été détectés non plus : un mapping de catégorie 1 entre la propriété *provider* du concept *appreciation* de l'ontologie et l'attribut *provider* de la DTD de CRT et un mapping de catégorie 5, pour la relation *is_appreciated* reliant les concepts *Cultural_object* et *appreciation*.

L'expérimentation montre que *DétectHybrideMap* a réussi à détecter 21 mappings avec les techniques automatiques de catégorie 1, 2 et 5, et 25 mappings conditionnels avec la technique 3 en interaction avec l'utilisateur, ce qui fait au total 46 mappings sur 51 (90 %). Ce résultat est encourageant.

Enfin, nous avons mesuré le temps d'exécution total du programme en utilisant deux machines différentes. Celui-ci se situait entre 21 et 30 secondes selon le type de serveur utilisé, un temps que nous avons jugé tout à fait satisfaisant.

7. Travaux proches

Beaucoup de données stockées dans des formats différents sont disponibles actuellement dans des sources distinctes et hétérogènes. Comme le nombre de systèmes utilisant ces données hétérogènes croît, il devient très important de concevoir des mécanismes de conversion ou de traduction de ces données d'un format dans un autre.

Les systèmes d'intégration d'information à base de médiateur sont confrontés à ce problème. En effet, les requêtes sont posées dans le vocabulaire d'une ontologie, ou schéma global, puis réécrites dans les termes des sources interrogées. L'interaction avec les sources d'information est alors assurée par des programmes interfaces, ou adaptateurs. A chaque source d'information est associé un tel programme qui accepte les requêtes auxquelles la source peut répondre, accède à la source, extrait les données et les retourne dans le format souhaité. Ces travaux ne se préoccupent pas du stockage des données retournées. Pour la plupart (Levy *et al*., 1996) (Hammer *et al*., 1997, Lehti *et al*., 2004, Miklos *et al*., 2005, Amann *et al*., 2002, Benatallah *et al*., 2005, Chen *et al*., 2006), ils se sont intéressés (1) au choix des langages utilisés pour modéliser le schéma global, ainsi qu'au choix des langages pour modéliser, en fonction de ce schéma, les vues sur les sources à intégrer et les requêtes des utilisateurs, (2) et, en fonction de ces choix de modélisation, à la conception et la mise en œuvre d'algorithmes de réécriture de requêtes en termes de vues pour le calcul des plans de requêtes à exécuter. Dans PICSEL, l'étude de ces problèmes a fait l'objet de différents travaux antérieurs (Rousset *et al*., 2002, Goasdoue *et al*., 2004).

Les médiateurs se distinguent par les langages utilisés mais aussi par la façon dont est établie la correspondance entre le schéma global et les schémas des sources de données à intégrer. On distingue l'approche GAV qui consiste à définir le schéma global en fonction des schémas des sources de données à intégrer. Comme les requêtes des utilisateurs sont exprimées en termes du schéma global, cette approche permet de facilement obtenir une requête en termes des schémas des sources. L'approche LAV est l'approche duale. Elle consiste à définir les schémas des sources de données, par un ensemble de vues, en fonction du schéma global. Cette deuxième approche est très flexible, contrairement à la précédente, pour l'ajout (ou la suppression) de sources de données à intégrer car cela n'a aucun effet sur le schéma global, seules des vues doivent être ajoutées. En revanche, la traduction d'une requête du schéma global en requêtes sur le schéma des sources demande un mécanisme de réécriture de requêtes complexe. Il existe des systèmes, tels SWIM (Koffina *et al*., 2006) adoptant une approche GLAV permettant de mettre en correspondance des éléments de l'ontologie avec des éléments de schémas de sources. Ces systèmes relèvent à la fois des approches GAV et LAV et tirent ainsi profit de leurs avantages respectifs.

Nos travaux se situent également dans le cadre de l'approche GLAV. Toutefois, SWIM s'appuie sur des mappings qu'il suppose connus. Dans PICSEL, la réécriture des requêtes est effectuée en termes de vues qui décrivent le contenu des sources. Notre objectif est de générer ces vues de façon automatique à partir de mappings entre l'ontologie et la source à intégrer, mappings générés également de manière automatique.

Des travaux proches du contenu de cet article se trouvent également dans le domaine des bases de données. Dans cette communauté, les travaux de traduction de données se distinguent des travaux portant sur la transformation de schémas (Batini *et al*., 1986) en ce sens qu'ils supposent que les schémas source et cible sont donnés.

Une différence est également faite avec l'intégration de schémas vue comme l'intégration d'un ensemble de schémas dans une représentation unifiée. Ces travaux consistent à créer un schéma intégré avant de créer des mappings entre les schémas. L'objectif de la traduction de données est de traduire le plus automatiquement possible les données associées à un schéma source dans le format des données d'un schéma cible. Beaucoup de travaux de traduction de données proposent des techniques portant directement sur les données (Abiteboul *et al*., 1997). D'autres permettent la représentation de schémas mais nécessitent l'écriture manuelle du programme de traduction (Cluet *et al.,* 1998). La technique proposée dans (Milo *et al*., 1998) est l'une des premières à exploiter l'information au niveau du schéma source pour aider à automatiser la traduction de données. Cette technique part de l'hypothèse que le schéma du système cible est souvent semblable à celui du système source. L'approche consiste à concevoir un système de traduction de données hétérogènes, appelé *TranScm,* basé sur la mise en correspondance de schémas, en utilisant une méthode à base de règles. Des règles de matching sont pré-définies initialement dans le système. Elles spécifient les mises en correspondances, ou mappings logiques, entre deux schémas sur la base de leurs propriétés communes. Ces règles peuvent être complétées par l'utilisateur pour prendre en compte les spécificités du schéma source (censées être mineures). La traduction des données est ensuite réalisée automatiquement.

En revanche, lorsque les schémas source et cible sont différents du point de vue de leur structure, les règles de matching ne peuvent plus être pré-définies. Ainsi, dans le système Clio (Popa *et al*., 2002), c'est l'utilisateur qui initie le processus en indiquant les mises en correspondance à établir entre les schémas. Cette spécification est toutefois grossière. L'utilisateur ne précise pas les conditions selon lesquelles les mappings sont valides. Le système affine la représentation des mappings déclarés en raisonnant sur les schémas pour intégrer les contraintes sémantiques qui y sont représentées. Des propositions de mappings sont faites à l'utilisateur qui peut avoir à choisir entre différentes alternatives. Ce processus semi-automatique est suivi d'une phase de traduction automatique des mappings logiques préalablement obtenus en requêtes sur les schémas sources afin d'obtenir les données satisfaisant les contraintes et la structure du schéma cible. Cette approche est centrée sur la représentation des mappings sémantiques effectuée sous forme de règles, la phase de traduction correspondant surtout à la traduction de ces règles dans le langage accepté par la source (XQuery pour une source XML, SQL pour une base de données relationnelle) et à leur exécution. De la même façon, le système MAPONTO (An *et al*., 2006) s'appuie également sur l'utilisateur pour obtenir des correspondances simples entre des éléments atomiques. Un raisonnement mis en œuvre sur ces correspondances permet ensuite de suggérer des règles de mappings complexes que l'utilisateur doit valider. Si la génération automatique des mappings est un point commun entre ces travaux et les nôtres, l'objectif recherché est différent. En bases de données, les mappings recherchés peuvent être complexes (peuvent nécessiter d'agréger des éléments), c'est pourquoi le processus de génération automatisée, quand il existe, prend appui sur l'utilisateur. Dans PICSEL,

les mappings utilisés sont simples. Il s'agit de relation 1 :1 entre des éléments de deux schémas, ce qui explique que l'automatisation peut être plus poussée. En revanche, les mappings dans PICSEL ne sont pas considérés comme une fin en soi. Ils servent à générer automatiquement les vues décrivant de façon synthétique le contenu des sources.

Enfin, nos travaux peuvent être rapprochés de travaux réalisés dans le domaine de l'alignement de schémas ou d'ontologies. Il existe de nombreux travaux visant à automatiser la génération de mappings mais déconnectés du problème de l'extraction de données. Une synthèse est présentée dans (Rahm *et al.,*2001), (Kalfoglou *et al.*, 2003), (Shvaiko *et al.*, 2005).

Nos techniques de recherche de mises en correspondance entre éléments sont originales par rapport à celles proposées dans la littérature. En effet, les techniques usuelles qu'elles soient terminologiques, structurelles ou sémantiques (Madhavan *et al*., 2001, Yan *et al*., 2001, Do *et al.*, 2001, Bach *et al*., 2004, Giunchiglia *et al.*, 2004, Kalfoglou *et al.*, 2005) ne conviennent pas car, à des concepts identiques correspondent soit des termes syntaxiquement identiques, ceux de l'ontologie de référence, soit des termes syntaxiquement très différents lorsque aucun équivalent n'existe dans l'ontologie.

Nous proposons principalement deux techniques. L'une exploite les mappings déjà établis avec l'ontologie et d'autres schémas de sources pour trouver les éléments du schéma de la nouvelle source les plus proches de ceux des schémas déjà mis en correspondance avec l'ontologie. Un rapprochement pourrait être fait avec les techniques basées sur l'utilisation de ressources complémentaires (Aleksovski *et al*., 2006, Giunchiglia *et al*., 2006, Sabou *et al.*, 2006) dont le recours est néanmoins délicat car il est toujours difficile de bien identifier le contexte d'interprétation des concepts provenant de ces ressources. Dans notre cas, ce problème de contexte à identifier n'existe pas. Les mappings déjà établis et réutilisés sont supposés avoir été validés. L'autre technique est spécifique aux documents XML. Elle prend en compte le fait que les schémas n'étant pas tous représentés au même niveau d'abstraction, il est possible de proposer un mode de représentation et d'identification des mappings dits conditionnels. L'accent est tout particulièrement mis sur la nécessaire exploitation des schémas des modèles comparés et des données associées, sans tirer partie de techniques d'apprentissage automatique comme cela est fait dans (Doan *et al*., 2003).

De façon plus générale, notre approche présente des ressemblances avec les travaux cités ci-dessus mais s'en distingue aussi pour plusieurs raisons.

Tout d'abord, elle combine la découverte de mappings avec le processus d'extraction et de traduction de données. Dans tous les travaux cités ci-dessus, l'accent est mis soit sur la découverte de mappings, soit sur l'extraction et/ou la traduction de données mais aucun ne combine les deux aspects du problème. Nous proposons une approche applicable au-delà du cadre applicatif ayant motivé nos travaux. Les applications visées sont celles centrées autour d'une ontologie de référence faisant inter-opérer différentes sources XML partageant une grande part du vocabulaire de cette ontologie mais hétérogènes du point de vue de leur structure.

L'approche se veut être la plus automatique possible (l'expert n'intervient que pour valider des propositions du système). Les techniques que nous proposons sont toutes guidées par l'ontologie. Cela permet de rechercher et de définir des mises en correspondance uniquement avec les seuls éléments de l'ontologie qui sont supposés être les seuls éléments intéressants sémantiquement. Cela évite de rechercher des mappings avec des éléments non signifiants des schémas à rapprocher. Les mises en correspondance établies sont de type 1 :1. A un élément de l'ontologie correspond (éventuellement) un élément, au sens général du terme, d'un schéma, cet élément pouvant être composé de différents sous-éléments dans la DTD. C'est sur cette base que sont définies les vues selon notre approche. Seules les propriétés et les relations exprimées dans l'ontologie sont introduites dans la vue relative à un concept. Les éléments d'une DTD non pertinents sémantiquement, car non mis en correspondance avec des éléments de l'ontologie, ne sont pas considérés. Notons enfin que les connaissances représentées dans l'ontologie sont supposées être des connaissances de base, ou élémentaires, à partir desquelles il est tout à fait possible de définir des connaissances plus complexes faisant intervenir, par exemple, des jointures. Si on suppose l'existence de telles définitions, le système que nous proposons peut parfaitement, sur la base des mappings des éléments mis en jeu, extraire les données correspondant à des connaissances complexes, ce qui est très intéressant.

Le dernier point distinctif de notre approche provient du fait qu'elle est parfaitement intégrée au système auquel elle s'applique, tout en restant générique. Elle ne conduit pas à introduire de nouveaux langages. Elle s'appuie sur des modules logiciels ayant plusieurs usages dans le cadre du système. Ainsi, le module de description du contenu des sources donne une description d'un schéma source en termes du schéma de l'ontologie (un sous-schéma en quelque sorte). Ce module est utile pour définir les vues exploitées et pour gérer les sources dont les données restent stockées localement. Il est aussi utilisé pour guider l'extraction des données qui seront stockées dans l'entrepôt local du système. Enfin, l'extraction de données fournit directement les données conformes au schéma de l'entrepôt, dans le bon format de sortie (RDF/XML), les requêtes XQuery permettant d'extraire les données étant générées à partir de termes de l'ontologie. Ceci s'oppose aux travaux qui nécessitent une phase de réécriture des données avant leur chargement dans l'entrepôt.

8. Conclusion et perspectives

Nous avons présenté une méthode unifiée permettant à la fois l'intégration d'une source externe via une approche de médiation et l'intégration de données dans un entrepôt local. Notre approche, applicable à des sources XML, combine découverte de mappings et construction de wrappers. Elle est générique, c'est-à-dire applicable quel que soit le domaine d'application, à des sources XML valides par rapport à une certaine DTD, à intégrer dans un système composé d'une ontologie représenté en RDF-S et d'un entrepôt de données RDF. Les mappings obtenus sont exprimés en

XPath et représentés de façon déclarative. Leur définition est ainsi tout à fait indépendante du processus d'extraction des données. Cela peut permettre d'enrichir ces mises en correspondance si nécessaire ou de les consulter et éventuellement les modifier. Le module d'extraction est capable d'extraire les données de tout document XML conformément à des mappings exprimés en XPath et par rapport à une certaine ontologie qui joue aussi le rôle de schéma de l'entrepôt dans lequel les données extraites sont stockées. Ce travail, réalisé dans le cadre du projet PICSEL3, devra être interfacé avec le module de réconciliation de références développé en parallèle dans l'équipe (Saïs *et al.*, 2007). Les données extraites d'une nouvelle source peuvent en effet porter sur des objets déjà représentés dans l'entrepôt, sous une forme qui peut ne pas être exactement identique. Il est nécessaire de repérer de telles situations afin d'éliminer toute redondance.

9. Bibliographie

Abiteboul S., Cluet S., Milo T., « Correspondence and translation for heterogeneous data », *Proceedings of ICDT*, 1997, p. 351-363.

Aleksovski Z., Klein M., Ten Kate W., Van Harmelen F., « Matching unstructured Vocabularies using a Background Ontology », *Proceedings of* EKAW, 2006, p. 182-197.

Amann B., Beeri C., Fundulaki I., Scholl M., « Querying XML sources using an ontology-based mediator », *Proceedings of CoopIS,* 2002, Irvine, California, p. 429-448.

An Y., Mylopoulos J., Borgida A., « Building Semantic mappings from databases to ontologies », *Proceedings of AAAI,* 2006*,* Boston, MA.

Bach T.-L., Dieng-Kuntz R., Gandon F., « On Ontology Matching Problems for building a Corporate Semantic Web in a Multi-communities Organization »*, Proceedings of ICEIS,* 2004, Porto, Portugal, p. 236-243.

Batini C., Lenzerini M., Navathe S.B., « A comparative analysis of methodologies for database schema integration »*, ACM Computing Surveys*, 1986, 18(4), p. 323-364.

Benatallah B. Hacid M., Paik H., Rey C., Toumani F. « Towards Semantic-driven, flexible and Scalable Framework for Peering and Querying e-catalog Communities »*, Information Systems International Journal, special issue on Semantic Web,* 2005.

Chen H., Wang Y., Wang H., Mao Y., Tang J., Zhou C., Yin A., Cui M., Wu Z., « From legacy Relational databases to the Semantic Web: an In-Use Application for Traditional Chinese Medecine. »*, Proceedings of ISWC,* 2006, Athens, USA, p. 750-763.

Cluet S., Delobel C., Simeon J., Smaga K., « Your mediators need data conversion! », *Proceedings of SIGMOD*, 1998, Seattle, USA, p. 177-188.

Do H. H., Rahm E., « COMA – A system for flexible combination of schema matching approaches »*, Proceedings of VLDB,* 2001, p. 610-621.

Giunchiglia F., Shvaiko P., Yatskevich M., « S-Match: an algorithm and an implementation of semantic matching »*, Proceedings of ESWC,* 2004, p. 61-75.

Giunchiglia F., Shvaiko P., Yatskevich M., « Discovering Missing Background Knowledge in Ontology Matching», *Proceedings of ECAI,* 2006, p. 382-386.

Goasdoue F., Rousset M.-C., « Answering Queries using Views: a KRDB Perspective for the Semantic Web », *ACM Journal Transactions On Internet Technology (TOIT),* Vol. 4, Issue 3, 2004, p. 255-288.

Hammer J., Garcia-Molina H., Nestorov S., Yerneni R., Breunig M., Vassalos V., « Template-Based wrappers in the TSIMMIS System », *Proceedings of SIGMOD,* 1997, Tucson, Arizona, p. 532-535.

Kalfoglou Y., Hu B., « CROSI Mapping System (CMS) Results of the 2005 Ontology Alignment Contest », *Integrating Ontologies Workshop, Proceedings of K-Cap,* 2005, Banff, Canada, p. 77-84.

Kalfoglou Y., Schorlemmer M., « Ontology mapping: the state of the art», *The Knowledge Engineering Review*, Vol. 18:1, 2003, p. 1-31.

Koffina I., Serfiotis G., Christophides V., Tannen V., « Mediating RDF/S Queries to Relational and XML Sources », *Int. J. Semantic Web Inf. Syst.* 2(4), 2006, p. 68-91.

Lehti P., Fankhauser P., « XML data Integration with OWL: Experiences and Challenges », *Proceedings of SAINT,* 2004,Tokyo, Japan, p. 160-170.

Levy A. Y., Rajaraman A., Ordille J. J., « Querying Heterogeneous Information Sources Using Source Descriptions », *Proceedings of VLDB*, 1996, Bombay, India, p. 251-262.

Madhavan J., Bernstein P. A., Rahm E., « Generic matching with Cupid », *International Journal of Very Large Data Bases,* 1(4): 334-350.

Miklos Z., Sobernig S., « Query translation between RDF and XML: A case study in the Educational Domain », *Proceedings of WWW Workshop on Interoperability of Web-Based Educational Systems,* 2005, Chiba, Japan, p.27-35.

Milo T., Zohar S., « Using Schema matching to Simplify Heterogeneous data translation », *Proceedings of VLDB*, 1998, p. 122-133.

Popa L., Velegrakis Y., Miller R. J., Hernandez M. A., Fagin R., « Translating Web data », *Proceedings of VLDB*, 2002, p. 598-609.

Rahm E., Bernstein P., « A survey of approaches to automatic schema matching », *International Journal of Very Large Data Bases,* 2001, 10(4), p. 334-350.

Rousset, M.-C., Bidault A., Froidevaux C., Gagliardi H., Goasdoue F., Reynaud C., Safar B., « Construction de médiateurs pour intégrer des sources d'information multiples et hétérogènes : le projet PICSEL », *Revue I3*, Vol. 2, Numéro 1, 2002, p. 9-59.

Sabou M., D'Aquin M., Motta E., « Using the Semantic Web as Background Knowledge for Ontology Mapping », *Proceedings of ISWC Workshop on Ontology Matching,* 2006, Athens, Georgia.

Saïs F., Pernelle N., Rousset M.-C., « L2R : a Logical method for Reference Reconciliation », *Proceedings of AAAI*, 2007,Vancouver, British Columbia, Canada, p.329-334.

Shvaiko, P., J. Euzenat, «A survey of Schema-based Matching Approaches », *Journal of Data Semantics IV*, 2005, p. 146-171.

Article reçu le xx xx 2007
Article accepté le xx juin 2008

Chantal Reynaud *est professeur en informatique à l'IUT d'Orsay – Université Paris-Sud et membre du LRI et du projet GEMO de l'INRIA Saclay Île-de-France. Ses activités de recherche se situent dans le domaine du Web sémantique et s'articulent autour de deux thèmes principaux: la médiation pour le Web sémantique, la recherche d'information sur le Web à base d'ontologies.*

Brigitte Safar *est maître de conférences en informatique à l'université Paris-Sud et membre de l'équipe-projet IASI-GEMO commune au LRI et à l'INRIA Saclay Île-de-France. Ses activités de recherche portent sur l'intégration d'informations et le Web sémantique.*

Annexe *Diagramme de classes UML représentant l'ontologie*